Alignement d'ontologies dirigé par la structure

Jean-François Djoufak-Kengue, Jérôme Euzenat, Petko Valtchev

Alignement d'ontologies dirigé par la structure

Jean François Djoufak Kengue*, Jérôme Euzenat** et Petko Valtchev*

*LATECE, Université du Québec à Montréal
djoufak.jean_francois@courrier.uqam.ca, valtchev.petko@uqam.ca
**INRIA Rhône-Alpes & LIG, Montbonnot, France
jerome.euzenat@inrialpes.fr

Résumé. L'alignement d'ontologies met en évidence les relations sémantiques entre les entités de deux ontologies à confronter. L'outil de choix pour l'alignement est une mesure de similarité sur les couples d'entités. Certaines méthodes d'alignement performantes font dépendre la similarité d'un couple de celles des couples voisins. La circularité dans les définitions résultantes est traitée par le calcul itératif d'un point fixe. Nous proposons un cadre unificateur, appelé *alignement dirigé par la structure*, qui permet de décrire ces méthodes en dépit de divergences d'ordre technique. Celui-ci combine l'appariement de graphes et le calcul matriciel. Nous présentons son application à la ré-implémentation de l'algorithme OLA, baptisée OLA_2.

1 Introduction

Une ontologie permet de décrire un domaine en définissant son vocabulaire et les axiomes qui le régissent. Les travaux actuels utilisent pour ce faire des formalismes proches des objets comme les logiques de descriptions (Baader et al., 2003). Dans ce cadre, une ontologie définit un ensemble de concepts et les relations qu'ils entretiennent avec d'autres concepts par spécialisation ou au travers de propriétés. Ainsi, les diagrammes UML de la figure 1 peuvent être considérés comme des (fragments d') ontologies.

Dans de nombreux contextes applicatifs, tel que l'échange de documents sur le Web, plusieurs ontologies couvrant, totalement ou partiellement, un même domaine se trouvent impliquées. Pour permettre l'interopérabilité des applications et/ou agents qui s'appuient sur une des ces ontologies, l'hétérogénéité entre les connaissances exprimées au sein de chacune d'entre elles doit être résolue. À cette fin, les liens sémantiques entre entités appartenant à deux ontologies différentes doivent être établis, c'est le but de l'*alignement d'ontologies* (Euzenat et Shvaiko, 2007).

Étant données deux ontologies, l'alignement produit un ensemble de correspondances chacune liant deux entités (par exemple, des concepts, des instances, des propriétés, des termes, etc.) par une relation (équivalence, subsomption, incompatibilité, etc.), éventuellement munie d'un degré de confiance. L'ensemble de correspondances, aussi appelé *alignement*, peut par la suite être utilisé pour fusionner les ontologies, migrer des données entre ontologies ou traduire des requêtes formulées en fonction d'une ontologie vers une autre.

Le nombre de méthodes d'alignement progresse constamment et une évaluation annuelle[1] permet les évaluer. Toutefois, le problème est loin d'être réglé, essentiellement à cause de sa nature exploratoire : l'alignement est une génération d'hypothèses sur la sémantique des entités en n'ayant accès qu'à une connaissance descriptive (d'ordre syntaxique). En conséquence, les méthodes vont typiquement appliquer des combinaisons de techniques élémentaires de nature heuristique (fouille de données, analyse de similarité, analyse de concepts, raisonnement formel, etc.), implémentées par des outils indépendants connu sous le nom de *matchers*. La contribution et l'orchestration de ces outils sont régies par un nombre important de paramètres qui sont spécifiques à chaque méthode. Ceci rend l'analyse de l'impact d'une technique, matcher ou paramètre, très difficile à évaluer. Malgré le nombre toujours croissant de méthodes et d'études comparatives dans le domaine, il manque encore un corpus de connaissance consensuelle sur la façon d'aborder les problèmes de l'alignement ainsi que sur les mérites respectifs des diverses techniques et méthodes. Aussi, l'exercice d'évaluation annuel OAEI est un pas dans la bonne direction, car il fournit un cadre équitable de comparaison entre méthodes.

Nous proposons une approche complémentaire pour appréhender les techniques d'alignement. En effet, à la place d'une comparaison basée sur la sortie des méthodes, c'est-à-dire, en mode "boîte noire", il est au moins aussi profitable d'aborder celles-ci en mode "boîte de verre". À cette fin, nous envisageons un cadre unificateur couvrant les méthodes existantes (et leur techniques) qui, lorsque proprement paramétré, se comporte comme une méthode concrète ou comme une autre. Les bénéfices d'un tel cadre sont nombreux, en commençant par la compréhension des mécanismes calculatoires à la base d'une méthode particulière. De plus, il permettra de clarifier les mérites respectifs des diverses composantes de la méthode en expérimentant plus finement les paramétrisations. Les résultats de telles études faciliteront à leur tour l'identification d'améliorations potentielles à apporter aux méthodes, et peuvent, à plus long terme, constituer les fondations d'une vraie théorie sur l'alignement.

Cependant, la conception d'un cadre unificateur pour l'ensemble des méthodes de la littérature semble irréaliste. Pour cette raison, nous proposons de réduire sa portée en se limitant à un groupe homogène de méthodes partageant certains principes et techniques. Ainsi, dans un premier temps, nous étudions un cadre inspiré de notre méthode OLA (Euzenat et Valtchev, 2004) qui couvre également GMO, utilisée dans le système Falcon (Jian et al., 2005), et SIMILARITY FLOODING ou SF (Melnik et al., 2002), utilisée dans le système Rondo. Deux de ces systèmes présentent de très bonnes performances lors des évaluations. Par ailleurs, ils ont tous trois ceci en commun qu'ils appuient la construction d'alignements sur les degrés de ressemblance entre entités dont la mesure de similarité sous-jacente est définie de façon récursive. En effet, les diverses définitions utilisées par ces méthodes traduisent toutes un principe simple énoncé déjà dans Bisson (1992), à savoir, que deux entités au sein d'une représentation structurée sont d'autant plus similaires que les entités qui les entourent le sont aussi. En conséquence, les valeurs pour les divers couples d'entités se retrouvent dans une dépendance mutuelle et récursive et par là ne peuvent être calculées qu'approximée par un processus itératif qui s'apparente à la transmission de similarité entre couples d'entités voisines.

Afin de capter cet aspect central dans les trois méthodes, nous proposons un cadre dit *d'alignement dirigé par la structure*, faisant ainsi référence au rôle de la structure relationnelle de chacune des ontologies à aligner en tant que conducteur de similarité. Le cadre repose sur la représentation des ontologies sous forme de graphes étiquetés ainsi que sur la modélisation des

[1] `http://oaei.ontologymatching.org`

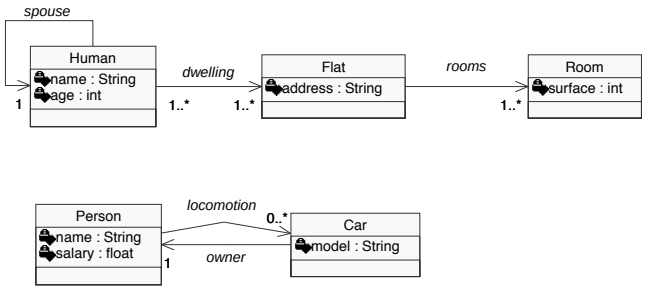

FIG. 1 – *Deux ontologies simples en* UML.

calculs itératifs à l'aide de produits de matrices. Les bénéfices de son utilisation se sont déjà fait observer : la ré-implémentation de OLA dans le nouveau cadre, a montré des performances supérieures à son ancêtre dans les exercices de l'OAEI.

Dans la suite, nous présentons d'abord les principes de la similarité par renforcement mutuel et soulignons les divergences entre les trois méthodes (§2). Ensuite, les bases de l'alignement dirigé par la structure sont présentées (§3). L'instanciation de celui-ci en suivant le schéma de OLA, résultant en OLA_2, est ensuite décrite (§4). Finalement, les résultats de l'évaluation annuelle OAEI-2007 sont rapportés (§5).

2 Alignement d'ontologies et modèles de similarité par renforcement mutuel

Nous présentons ci-dessous les principes généraux sur lesquels est bâti OLA en indiquant les nombreuses différences avec les algorithmes GMO et SF. Les trois algorithmes étudiés confrontent les ontologies à l'aide d'une mesure de similarité intégrant toutes les facettes des entités au sein de l'ontologie. Ils traduisent un principe de renforcement mutuel entre entités reliées, introduit par (Bisson, 1992) et adapté dans (Valtchev, 1999).

2.1 Le problème de l'alignement

Une ontologie décrit les concepts intervenant dans un domaine particulier. Ces concepts sont typiquement organisés dans une hiérarchie de généralisation et peuvent être reliés par des relations, ou propriétés. Au lieu de relier un concept à d'autres concepts, certaines relations peuvent avoir comme co-domaine des types de données standards. Des objets, c'est-à-dire, des instances des concepts, peuvent également être présents, eux-mêmes connectés par des instances de propriétés. Concernant la description d'une ontologie, le langage OWL (Dean et Schreiber (eds.), 2004) est maintenant un standard. Néanmoins, de nombreuses ontologies sont exprimées dans d'autres formalismes.

À titre d'illustration, la figure 1 présente deux ontologies simples, O_1 (haut) et O_2 (bas), visualisées comme des diagrammes de classes UML. Ici, les concepts ontologiques correspondent à des classes UML et les propriétés ontologiques à des variables de classe ou bien à des rôles des associations.

Le but de l'alignement est d'expliciter les éventuels rapports sémantiques entre les entités (concepts, relations, instances) appartenant à des ontologies différentes. Dans notre exemple, cela signifie de pouvoir affirmer que les classes `Person` et `Human` dans la figure 1 sont équivalentes tout comme les deux attributs `name`s. Au contraire, `Person` et `Car` sont disjoints en tant que classes.

La confrontation, deux-à-deux, des entités qui a pour but de leur affecter un degré "d'alignabilité" peut être faite de nombreuses façons et en se basant sur des aspects variés. Par exemple, pour deux classes UML, une ou plusieurs dimensions descriptives, tels leurs noms, leurs instances connues (dans une base de données), les super-classes respectives, les listes de variables d'instances, les rôles d'associations, etc., peuvent être choisies. À leur tour, les rôles d'association pourront être comparés en fonction de leurs noms, les classes propriétaires et les classes co-domaines respectifs, les cardinalités, etc. De façon générale, à des fins de comparaison, les ontlogies peuvent-être vues comme des graphes dont les nœuds sont des entitiés des divers langages (classes, instances, propriétés) et les arcs sont typés par des catégories de relations (instanciation, spécialisaton, attribution).

La définition d'une mesure de similarité sur une telle représentation sera donc une expression combinant (par exemple, dans une somme ou une combinaison linéaire) les similarités de chacune des dimensions descriptives choisies pour l'alignement. Par exemple, pour les classes, nous pouvons choisir de ne travailler que sur les noms et les attributs, ignorant les rôles et les super-classes éventuelles. La comparaison sur chacune des dimensions requiert une fonction dédiée. Dans le cas des noms des entités, de nombreuses possibilités existent dont la plus simple est la comparaison de ceux-ci en tant que chaînes de caractères. Cependant, une telle comparaison risque d'être parfois trompeuse comme le montre l'exemple de `Person` et `Human`. Une approche alternative consiste à utiliser un thesaurus ou autre ressource lexicale afin de capter les rapports entre termes (synonymie, homonymie, hypo-/hypero-nymie, etc.). Bien que plus puissante, elle se heurte aussi à l'ambiguïté intrinsèque de la langue, d'où l'impossibilité de limiter la confrontation aux simples noms.

2.2 Similarité par renforcement mutuel

Nous parlons de renforcement mutuel lorsque la similarité implémente le principe suivant : deux entités sont d'autant plus similaires que les entités directement reliées à celles-ci le sont aussi.

Dans un tel modèle de similarité, les deux questions posées ci-dessus reçoivent des réponses immédiates. Ainsi, d'une part, toutes les dimensions de la description d'une méta-classe entité participent dans l'expression de la similarité avec leur propre mesure de similarité. Par exemple, la similarité de classes appliquée à `Flat` et `Person`, soit $sim_C(\texttt{Flat}, \texttt{Person})$, va dépendre *a priori* de la similarité d'attributs sim_A pour tous les couples mixtes d'attributs, en particulier $sim_A(\texttt{address}, \texttt{salary})$. La dépendance effective entre $sim_C(\texttt{Flat}, \texttt{Person})$ et $sim_A(\texttt{address}, \texttt{salary})$ peut toutefois être sujette à des paramétrisations telle que la pondération ou la pré-sélection (voir ci-dessous).

De plus, même si cela n'est pas souvent affirmé de façon explicite, *toutes* les dimensions descriptives participent *a priori* à l'expression de la similarité entre deux entités. Ainsi, $sim_C(\texttt{Flat}, \texttt{Person})$ va dépendre non seulement des attributs respectifs mais surtout, et avant tout autre chose, de la similarité des termes $sim_\lambda(\texttt{'flat'}, \texttt{'person'})$. Les super-classes, absentes ici, et les rôles (ex., le couple (`rooms`,`locomotion`)) complètent la liste.

Une telle définition mène à une dépendance mutuelle entre valeurs de la similarité. Par exemple, chacune des deux valeurs $sim_C(\texttt{Flat}, \texttt{Person})$ et $sim_A(\texttt{address}, \texttt{salary})$ dépend de l'autre. En conséquence, le calcul des valeurs ne pourra pas toujours être effectué de façon directe et exacte. D'autres moyens doivent donc être déployés pour traiter la circularité. Avant de les aborder, précisons que la circularité dans les définitions peut être évitée soit en imposant des structures descriptives qui ne comportent pas de cycle, soit en utilisant deux fonctions différentes pour les entités, une en tant que couple comparé (ex., $sim_A(\texttt{address}, \texttt{salary})$) et une autre en tant que contributeurs à la similarité d'un couple voisin (ex., $sim_a(\texttt{address}, \texttt{salary})$).

Néanmoins, si les noms des entités contribuent effectivement à la similarité des entités, (tout comme les similarités de cardinalité, de types de données et de valeurs) les valeurs respectives constituent une partie calculable immédiatement de la valeur finale. Celle-ci reste stable au long des itérations décrites ci-dessous. Cette partie stable est un élément clé dans le modèle de similarité dans OLA, sa formation est optionnelle dans SF, mais dans GMO il n'existe pas car l'algorithme ne prend pas en compte les noms des entités. Ainsi, la valeur de similarité rendue par GMO ne reflète que la position de l'entité vue comme un nœud au sein du graphe de l'ontologie. En revanche, la similarité dans OLA et SF reflète également la concordance des deux noms et autre descripteurs locaux.

La différentiation dans les contributions des couples voisins est un autre mécanisme qui oppose OLA et SF à GMO. En effet, dans GMO, étant donné un couple, *tous* les couples possibles de voisins contribuent à la similarité du couple et ceci, avec le même poids. Dans SF, aucun couple n'est exclu, cependant, les poids sont partagés entre les couples voisins reliés par la même méta-association. Ainsi, les couples (`address`,`salary`) et (`address`,`name`) doivent partager un poids de 1 dans l'expression de $sim_C(\texttt{Flat}, \texttt{Person})$. Dans OLA, les contraintes sont encore plus sévères : pour des raisons de normalisation dans les valeurs définitives, il est même impossible que les deux couples ci-dessus contribuent simultanément à $sim_C(\texttt{Flat}, \texttt{Person})$. À tout instant, la contribution d'une des deux valeurs, $sim_A(\texttt{address}, \texttt{salary})$ ou $sim_A(\texttt{address}, \texttt{name})$, sera donc 0. Cependant, le couple dont la contribution est ignorée peut changer au long du processus de calcul. De façon générale, étant donné deux entités toutes deux possédant plus qu'un seul lien d'un type particulier (ex., `Human` et `Person` ont chacune deux attributs), OLA cherchera à apparier les entités sous-jacentes de façon optimale, c'est-à-dire, en maximisant la similarité totale.

2.3 Calculer les valeurs de la similarité

Le calcul des valeurs de la similarité suit le même schéma dans les trois systèmes. Au départ, les similarités sont initialisées, ensuite, de façon itérative, elles sont recalculées en utilisant, à une étape $k+1$, les valeurs des couples voisins pour l'étape k.

Dans GMO, l'initialisation est uniforme et tout couple reçoit une valeur de 1 au début. OLA et SF utilisent pour cela les similarités "locales" (noms, cardinalités, etc.). Néanmoins, la mise à jour itérative des valeurs diverge par la suite.

Dans SF et GMO, les valeurs de l'étape k sont remplacées par la somme, éventuellement pondérée dans SF, des valeurs des voisins. Ainsi, si nous imaginons une telle étape, la similarité $sim_C(\texttt{Flat}, \texttt{Person})$ serait égale à la somme des similarités des attributs et des rôles dans GMO, SF ajoutera à cette somme la similarité des noms.

Une fois toutes les valeurs de l'étape courante calculées, SF et GMO normalisent ces dernières en les divisant par une quantité appropriée. Pour GMO, la normalisation est obtenue

au prix d'une division par la norme euclidienne de la matrice des similarités. C'est une valeur bien plus grande que le maximum parmi toutes les valeurs calculées à l'étape courante qui est utilisé par SF.

OLA ne fait pas de calculs explicites pour forcer la normalisation. Celle-ci est est assurée par les formules utilisées dans le re-calcul des similarités. En revanche, avant de procéder à la mise à jour, la méthode doit établir un appariement maximal par couple et par type de lien. Tout comme pour les mises à jour, dans la production de ces appariements, les valeurs de l'étape précédente sont utilisées.

L'objectif du calcul dans les trois cas est d'atteindre par des pas successifs un point fixe, cependant, pour SF ceci n'est pas automatiquement assuré. Le calcul peut donc être interrompu après un nombre d'itérations pré-défini. OLA et GMO convergent vers des points-fixes.

2.4 Propriétés de la mesure et du processus itératif

Dans OLA, le calcul converge car les valeurs successives de chaque couple d'entités sont croissantes et bornées par 1. Ceci est imposé par le modèle (somme des poids égale à 1, normalisation, appariements, etc.). GMO converge par le poids de la norme euclidienne.

En plus de la convergence, OLA calcule une fonction de similarité qui est *maximale* : une fois lancé sur deux copies identiques de la même ontologie, la méthode est capable d'évaluer correctement à 1 la similarité des couples de copies de la même entité.

3 Alignement dirigé par la structure

Pour représenter d'une manière unifiée et formelle la famille d'algorithmes d'alignement dont OLA, GMO et SIMILARITY FLOODING sont les représentants, nous proposons le cadre d'alignement dirigé par la structure. Nous le présentons ici par ses deux composants graphiques (§3.1.2) et son composant calculatoire (§3.2).

3.1 Composants graphiques

Dans cette section, nous considérons une ontologie comme un ensemble d'entités (objets, classes, propriétés, etc.) et de liens sémantiques (instanciation, subsomption, équivalence, etc.) entre ces entités. Pour pouvoir être alignées, les ontologies sont au préalable représentées sous forme de graphes d'ontologie (§3.1.1). Ces derniers graphes sont ensuite utilisés pour construire des graphes de similarité (§3.1.2) qui représentent aussi bien les correspondances potentielles entre entités que les influences que certaines correspondances ont sur les autres.

3.1.1 Graphe d'ontologie

Le graphe d'ontologie est un graphe orienté et étiqueté. Ses nœuds représentent des entités de l'ontologie et possèdent deux étiquettes. L'une des étiquettes de nœud est le nom de l'entité représentée alors que l'autre identifie la catégorie de ladite entité. Comme catégorie d'entité on peut avoir CLASSE, OBJET, RELATION, PROPRIÉTÉ, etc.

Deux nœuds du graphe d'ontologie sont reliés par un arc si et seulement s'il existe un lien sémantique entre les entités qu'ils représentent. Chaque arc est étiqueté par le nom du lien qu'il

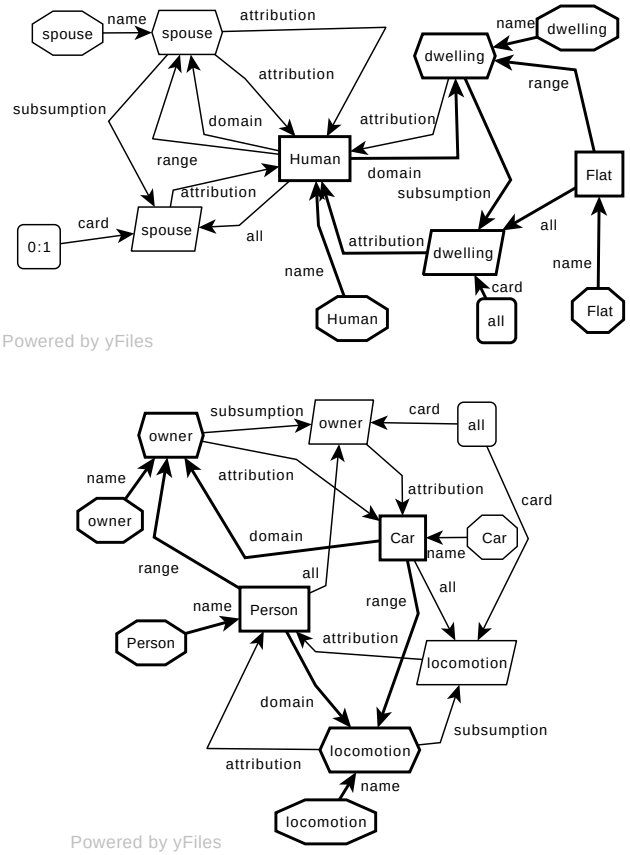

FIG. 2 – *Parties de graphes d'ontologie encodant les ontologies correspondants aux modèles UML de la figure 1.*

encode. Dans cet article, nous traitons de l'alignement des ontologies qui partagent le même ensemble de liens. Ce qui est le cas quand les ontologies sont définies dans le même langage.

La figure 2 présente des parties de graphes d'ontologie construits par OLA_2 pour représenter les ontologies correspondants aux modèles UML de la figure 1. Dans cette figure, les nœuds représentent des entités d'ontologie. La forme de chaque nœud dénote sa deuxième étiquette, c'est-à-dire la catégorie de l'entité représentée par le nœud. C'est ainsi que rectangles, parallélogrammes, hexagones, octogones et rectangles à bord arrondis représentent respectivement des entités qui sont des classes, des propriétés, des relations, des mots et des cardinalités. Les arcs représentent les liens entre entités et sont étiquetés par les noms desdits liens. Les parties en gras seront utilisées à la section suivante pour illustrer le graphe de similarité.

3.1.2 Graphe de similarité

Le graphe de similarité est un graphe orienté, étiqueté et pondéré. Il est le cœur de l'alignement dans la mesure où il produit la matrice d'adjacence qui est nécessaire au calcul de la similarité.

Soient G_1 et G_2 deux graphes d'ontologie et soit S le graphe de similarité correspondant. Les nœuds de S sont des couples $(v_1, v_2) \in G_1 \times G_2$ de nœuds tels que v_1 et v_2 représentent des entités appartenant à la même catégorie. La catégorie du nœud (v_1, v_2) est la catégorie commune de ses composantes. Contrairement aux nœuds du graphe d'ontologie, on associe aux nœuds du graphe de similarité une valeur représentant leur similarité.

Un arc d'étiquette l existe entre deux nœuds (v_1, v_2) et (v_1', v_2') du graphe de similarité si et seulement s'il existe simultanément un arc d'étiquette l entre les nœuds v_1 et v_1' de G_1 d'une part et les nœuds v_2 et v_2' de G_2 d'autre part. Les arcs du graphe de similarité sont pondérés conformément à la méthode d'alignement pour lesquels ils sont construits. Si le nœud (v_1', v_2') contribue à la similarité/dissimilarité du nœud (v_1, v_2) suivant le lien l qui lie les entités représentées, l'arc d'étiquette l existant entre (v_1', v_2') et (v_1, v_2) est orienté du premier nœud vers le deuxième.

L'ensemble des nœuds du graphe de similarité représente ainsi l'ensemble de tous les couples d'entités qui sont potentiellement mis en alignement. En effet, d'un point de vu purement structure, deux nœuds sont alignables si les entités qu'ils représentent appartiennent à la même catégorie et partagent le même ensemble de liens.

Le graphe de similarité construit à partir des parties en gras des graphes d'ontologie de la figure 2 est donné dans la figure 3. Dans les nœuds cette figure, la première (resp. deuxième)

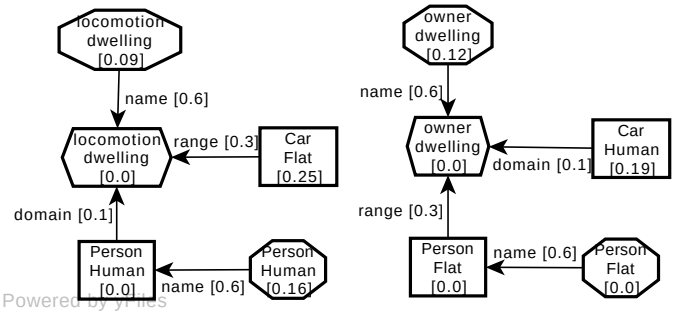

FIG. 3 – *Graphe de similarité construit à partir des parties en gras des graphes d'ontologie de la figure 2.*

ligne est une étiquette en provenance du graphe d'ontologie de droite (resp. gauche) de la figure 2. Les poids initiaux des nœuds et ceux des arcs sont indiqués entre crochets. Les arcs étiquetés *attribution*, *subsumption*, *card* et *all* ne se retrouvent pas dans le graphe de similarité puisqu'ils ne sont pas partagés par les parties en gras de la figure 2. Un raisonnement similaire s'applique en ce qui concerne la catégorie du nœud étiqueté *all* dans la partie gauche de la figure 2.

3.2 Composant calculatoire

Le calcul de similarité commence par affecter des valeurs initiales aux nœuds (v_1, v_2) du graphe de similarité en comparant les noms des entités représentées par les nœuds $v_1 \in G_1$ et $v_2 \in G_2$. La valeur associée à chaque nœud représente la similarité entre ces entités. Puis, à chaque itération cette valeur, est recalculée en fonction de celles des nœud adjacents obtenues à l'itération précédente. La contribution de chaque valeur est pondérée en la multipliant par le poids de l'arc considéré. L'ensemble des contributions est additionné pour donner la nouvelle valeur du nœud. Ce calcul de similarité est itéré jusqu'à ce que les valeurs des nœuds du graphe de similarité ne varient plus entre deux itérations successives. On dit qu'on a atteint le point fixe ou point de convergence.

Les algorithmes GMO et SF ne tiennent pas compte de l'orientation des arcs (la matrice d'adjacence est symétrique). En conséquence, ils réalisent la propagation dans les deux sens alors que l'algorithme OLA, qui tient compte de l'orientation des arcs, ne propage que dans un seul sens. Dans tous les cas, le calcul de similarité se réduit, comme montré dans (Blondel et al., 2004), à une équation matricielle ayant la forme suivante.

$$V_{i+1} = M_i \times V_i, i = 0, 1, 2, 3... \quad (1)$$

Dans cette équation, V_{i+1} (resp. V_i) est le vecteur des similarités des nœuds du graphe de similarité au début de l'étape $i + 1$ (resp. i) alors que la matrice M_i représente la matrice d'adjacence du graphe de similarité. En général, M_i ne change pas entre deux itérations, mais dans le cas de OLA, on verra à la section 4.3 que la matrice change pour refléter un appariement local. Le vecteur V_0 quand à lui, représente les poids initiaux des nœuds du graphe de similarité.

Le calcul de similarité ainsi modélisé réalise l'appariement de graphes. Sa convergence n'est cependant pas toujours acquise. En effet, il existe des cas dans lesquels l'algorithme SF ne converge pas. La convergence observée dans l'algorithme GMO prends ses racines dans (Blondel et al., 2004) alors que celle de l'algorithme OLA, décrite dans (Euzenat et Valtchev, 2004), est induite par des contraintes imposées lors de l'affectation des poids aux arcs du graphe de similarité.

4 Description de $\mathbf{OLA}_2$

Dans cette section nous présentons OLA_2 qui est une reformulation de l'algorithme OLA par rapport au cadre de l'alignement dirigé par la structure. Nous décrivons le graphe d'ontologie et le calcul du graphe de similarité (§4.1). Nous donnons ensuite des indications sur l'initialisation du vecteur de similarité et sur la matrice d'adjacence (§4.2). Par la suite, nous détaillons le processus de calcul de similarité (§4.3). Nous terminons par une vue globale de l'algorithme OLA_2 (§4.4).

4.1 Graphe d'ontologie et graphe de similarié

Graphe d'ontologie : Considérons les catégories de nœuds et les étiquettes des arcs manipulées par OLA. La figure 4 montre le méta-modèle qui est utilisé pour relier les nœuds du graphe d'ontologie en fonction de la catégorie des entités qu'ils représentent. Sur cette figure, les catégories sont représentées par des rectangles alors que les étiquettes sont associées aux

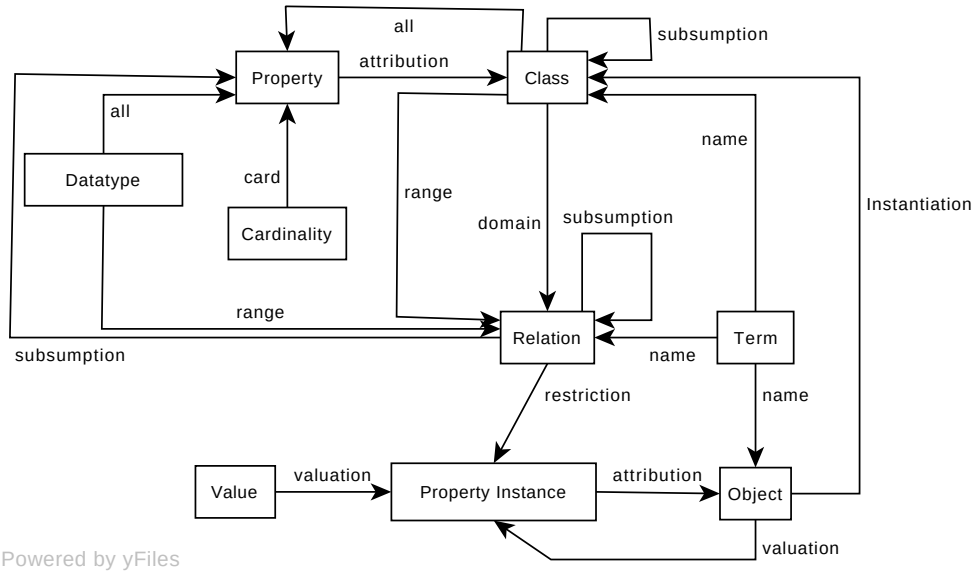

FIG. 4 – *Modéle utilisé par* OLA_2 *pour construire ses graphes d'ontologie.*

arcs. L'existence d'un arc nommé *instantiation* du rectangle étiqueté *object* au rectangle étiqueté *class* signifie qu'un lien *instanciation* ne peut lier qu'une entité de la catégorie OBJET (la source du lien) à une entité de la catégorie CLASSE (la cible du lien). L'interprétation des autres arcs de la figure 4 se fait de façon analogue.

Le graphe d'ontologie de OLA_2 est alors une instanciation du méta-modèle représenté par la figure 4. En effet, les nœuds du graphe d'ontologie sont des instances de nœuds de la figure 4 auxquels on associe une autre étiquette, le nom de l'entité représentée. De plus, les arcs du graphe d'ontologie sont des copies d'arcs de la figure 4.

Notons que l'existence d'un arc d'étiquette l d'une catégorie à l'autre signifie qu'il pourrait exister dans l'ontologie des entités de la première catégorie qui puissent être liées à des entités de la seconde catégorie par un lien nommé l. Au stade actuel, seuls les liens explicitement encodés dans les fichiers OWL représentant les ontologies sont pris en compte. La figure 2 présente deux parties de graphes d'ontologie produits par OLA_2.

Graphe de similarité : La construction du graphe de similarité est simple dès lors que les deux graphes d'ontologie sont connus. En effet, il suffit d'appliquer la procédure de la section 3.1.2 pour créer les nœuds et les arcs. Le graphe obtenu conserve les catégories des entités représentées par les nœuds du graphe d'ontologie, de même que les étiquettes et l'orientation des arcs. Cependant, le poids de chaque arc est affecté en fonction de son étiquette. Tous les arcs de même étiquette ont le même poids quelle que soit l'exécution de OLA_2 qui est considérée. Ces poids sont calculés par un processus externe à OLA_2 qui s'assure que les résultats fournis par ce dernier sur le jeu de test *benchmarks* d'OAEI sont les meilleurs possibles. Ce processus

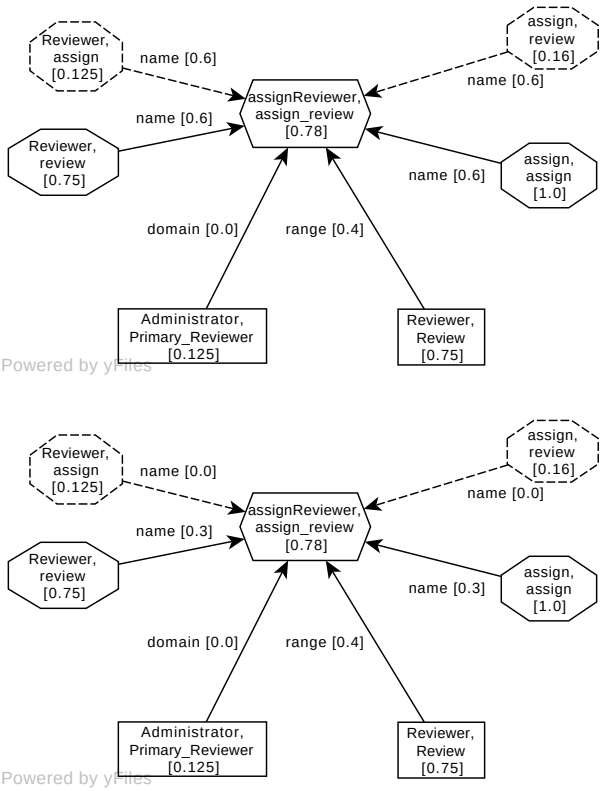

FIG. 5 – *Illustration de la sélection effectuée par* OLA_2 *lors du calcul de similarité.*

assure également que la somme des poids associés aux arcs incidents à une même catégorie de nœuds est 1.

4.2 Vecteur de similarité initial et matrice d'adjacence

Seuls les nœuds du graphe de similarité qui représentent des entités de catégorie MOT, TYPE DE DONNÉES, VALEUR et CARDINALITÉ se voient affecter des similarité initiales non nulles. Ces valeurs sont calculées au moyen d'une distance d'édition entre les deux étiquettes contenues dans les nœuds considérés. Les valeurs affectées aux nœuds de la figure 3 sont ces valeurs de similarité initiales.

La matrice d'adjacence représente l'influence de la similarité d'un nœud du graphe sur celle d'un autre nœud. Elle contient donc des valeurs entre 0 et 1, et non uniquement des valeurs booléennes comme une matrice d'adjacence classique. Ces valeurs seront utilisées pour pondérer la contribution de chaque nœud incident à la similarité d'un nœud. La matrice initiale M_0 est obtenue en retenant les valeurs figurant sur les arcs du graphe de similarité (figure 3), une absence d'arc étant noté par 0.

4.3 Calcul de similarité

La calcul itératif de la similarité de tous les nœuds du graphe de similarité s'exprime de façon uniforme par une équation matricielle similaire à celle présentée à la section 3.2. La spécificité de OLA_2 est que la matrice d'adjacence change à chaque itération. En effet, pour maximiser la similarité des nœuds du graphe de similarité, OLA_2 sélectionne avant chaque itération les arcs qui seront utilisés pour calculer la similarité.

Étant donné un nœud (v_1, v_2) du graphe de similarité, notons $\mathcal{N}_l(v_1, v_2)$ l'ensemble des nœuds qui sont la source des arcs d'étiquette l ayant le nœud (v_1, v_2) pour cible. D'après l'équation 1, les valeurs de similarité à l'étape i de tous les nœuds de $\mathcal{N}_l(v_1, v_2)$ sont utilisées pour calculer celles du nœud (v_1, v_2) à l'étape $i + 1$. Pour pouvoir maximiser la similarité, OLA_2 sélectionne, au moyen de l'algorithme d'appariement de (Hopcroft et Karp, 1973), les nœuds de $\mathcal{N}_l(v_1, v_2)$ dont les similarités vont être utilisées. Afin de forcer l'equation 1 à ne pas utiliser la similarité des nœuds non sélectionnés, les poids des arcs d'étiquette l qui les relient au nœud (v_1, v_2) prennent la valeur nulle, ce qui se traduit par une mise à jour des entrées correspondantes dans la matrice d'adjacence M_i. C'est ainsi que dans la partie supérieure de la figure 5, nœuds et arcs en traits discontinus ne sont pas sélectionnés pour le calcul de la similarité. Par la suite, les poids des arcs non sélectionnés prennent la valeur 0 alors qu'on divise les poids des arcs dont les nœuds source sont sélectionnés par une valeur de normalisation. Si $\mathcal{N}_l(v_1)$ et $\mathcal{N}_l(v_2)$ sont respectivement les ensembles de nœuds liés à v_1 et v_2 par des arcs d'étiquette l, la valeur de normalisation est $\max(|\mathcal{N}_l(v_1)|, |\mathcal{N}_l(v_2)|)$. Dans l'exemple de la figure 5, elle vaut 2 pour les arcs d'étiquette *name* et 1 pour les autres. En effet, dans les graphes d'ontologie utilisés pour construire le graphe de similarité dont une partie est illustrée à la figure 5, les nœud étiquetés *assignReviewer* et *assign_review* ont respectivement deux arcs étiquetés *name* et un arc étiqueté *domain* (resp. *range*). Ce qui explique les poids des arcs du graphe de dessous (figure 5).

4.4 Algorithme

Algorithmiquement, OLA_2 se présente comme séquence de cinq étapes. Les deux ontologies à aligner sont tout d'abord encodées sous forme de graphes d'ontologie en utilisant la fonction `OWL2OntologyGraph(.)`. Ces graphes d'ontologie sont ensuite utilisés par la fonction `ComputeSimilarityGraph(.,.)` pour construire le graphe de similarité. La matrice d'adjacence et le vecteur des similarités initiales sont alors respectivement calculés via les fonctions `ComputeAdjacencyMatrix(.)` et `ComputeInitialCoefficients(.)` à partir du graphe de similarité. On dispose dès lors de toutes les donnés requises pour exécuter la fonction `ComputationalComponentIteration(.,.)` qui implémente une itération du calcul de similarité. Mais, avant chaque itération, le vecteur des similarités est utilisé par la fonction `UpdateAdjacencyMatrix(.,.)` pour modifier la matrice d'adjacence. À la fin du calcul de similarité, les alignements sont produits par extraction de couplages maximaux au moyen de l'algorithme Hongrois (Munkres, 1957) implémenté dans la fonction `HungarianExtraction(.)`. Cette description correspond à l'algorithme 1.

Algorithm 1 : OLA_2

Entrée : $\mathcal{O}_1$ et $\mathcal{O}_2$, deux ontologies décrites en OWL
Sortie : $\mathcal{C}$, un ensemble de correspondances entre entités de $\mathcal{O}_1$ et $\mathcal{O}_2$
1 : G_1 = OWL2OntologyGraph($\mathcal{O}_1$) ;
2 : G_2 = OWL2OntologyGraph($\mathcal{O}_2$) ;
3 : S = ComputeSimilarityGraph(G_1, G_2) ;
4 : V = ComputeInitialCoefficients(S) ;
4 : $\mathcal{M}$ = ComputeAdjacencyMatrix(S) ;
5 : **repéter**
6 : $\mathcal{M}'$ = UpdateAdjacencyMatrix($\mathcal{M}$,V) ;
7 : V = ComputationalComponentIteration($\mathcal{M}'$,V) ;
8 : **jusqu'à convergence**
9 : $\mathcal{C}$ = HungarianExtraction(V) ;

5 Evaluation expérimentale de OLA_2

Nous avons participé à la dernière évaluation annuelle des méthodes d'alignement d'ontologies, OAEI-2007[2], avec l'algorithme OLA_2. Les résultats de l'évaluation sont publiés à l'url `http://oaei.ontologymatching.org/2007/results/`. OLA_2 n'a été appliqué que sur trois jeux de tests. Notamment : *benchmarks*, *conference* et *directory*, car OLA_2 avait de la peine à s'exécuter sur notre ordinateur d'expérimentation quand la taille des ontologies devenait assez grande. En effet, les produits matriciels coûtent cher en espace mémoire et en temps lorsque les matrices d'adjacence sont calculées à partir des graphes de similarité de grande taille. Une des extensions envisagées est de remédier à cette limitation.

Le jeu de test *benchmarks* est constitué d'ontologies engendrées automatiquement pour évaluer la robustesse des différents algorithmes. L'une des grandes difficultés de ce jeu de test est d'aligner des ontologies pour lesquelles les noms d'entités et parfois la structure n'ont aucune signification car ayant été anonymisés et/ou altérés par un processus aléatoire. Dans ces cas difficiles, OLA_2 a des meilleures moyennes de précision et rappel que celles de OLA. De même, OLA_2 présente des résultats nettement au dessus de ceux de bon nombre d'algorithmes évalués sur ce jeu de test. En effet, sa courbe de performance se trouve dans le groupe de tête.

S'agissant du jeu de test *conference*, il est constitué des ontologies hétérogènes qui représentent l'organisation de conférences et qui sont utilisées dans des applications réelles. OLA_2 termine avec 73.6% de précision et 44.5% de rappel. Il est également à noter que OLA_2 a un taux d'erreur de 6.7% pour la découverte erronée des liens de subsumption et des correspondances.

Enfin, le jeu de test *directory* est constitué de repertoires de sites web à aligner. OLA_2 y a dominé les autres participants. En effet, sa précision, son rappel et sa F-mesure sont au moins 1.5 fois meilleurs que ceux des systèmes concurrents et 2.6 fois meilleurs que les résultats produits par OLA lors d'OAEI-2005.

[2] `http://oaei.ontologymatching.org/2007/`

6 Conclusion

Nous avons présenté l'alignement dirigé par la structure, un cadre unificateur pour plusieurs méthodes d'alignement utilisant une similarité définie par renforcement mutuel. Motivé par un besoin de faciliter la compréhension de l'architecture fonctionnelle et du comportement des méthodes sur divers jeux de données pour l'alignement, le cadre utilise des formalismes de représentation et des mécanismes calculatoires bien connus, à savoir, graphes orientés étiquetés et calcul matriciel, respectivement. Nous avons illustré les bénéfices du nouveau cadre dans le cas de l'algorithme OLA dont la reformulation a donné naissance à OLA_2. L'intérêt de ce dernier a été confirmé par ses performances accrues comparées à celle de OLA.

Dans une prochaine étape, les deux autres algorithmes visés seront ré-implémentés au sein du cadre afin d'évaluer les gains potentiels. À part l'aspect crucial qu'est le calcul de la similarité, nous allons nous concentrer sur les mécanismes auxiliaires tels que l'affectation des poids, la comparaison des termes complexes et la normalisation.

Références

Baader, F., D. Calvanese, D. McGuinness, D. Nardi, et P. Patel-Schneider (Eds.) (2003). *The description logic handbook : theory, implementations and applications*. Cambridge University Press.

Bisson, G. (1992). Learning in FOL with similarity measure. In *Proc. 10th American Association for Artificial Intelligence conference, San-Jose (CA US)*, pp. 82–87.

Blondel, V. D., A. G. M. Heymans, P. Senellart, et P. V. Dooren (2004). A measure of similarity between graph vertices. with applications to synonym extraction and web searching. *SIAM Review 46*(4), 647–666.

Dean, M. et G. Schreiber (eds.) (2004). OWL web ontology language : reference. Recommendation, W3C. http ://www.w3.org/TR/owl-ref/.

Euzenat, J. et P. Shvaiko (2007). *Ontology matching*. Heidelberg (DE) : Springer-Verlag.

Euzenat, J. et P. Valtchev (2004). Similarity-based ontology alignment in OWL-Lite. In *European Conference on Artificial Intelligence ECAI-04*, pp. 333–337.

Hopcroft, J. E. et R. M. Karp (1973). An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal of Computing 2*(4), 225–231.

Jian, N., W. Hu, G. Cheng, et Y. Qu (2005). Falcon-AO : Aligning ontologies with falcon. In *Proceedings of the K-CAP Workshop on Integrating Ontologies 2005*, pp. 87–93.

Melnik, S., H. Garcia-Molina, et E. Rahm (2002). Similarity flooding : A versatile graph matching algorithm and its application to schema matching. In *ICDE '02 : Proceedings of the 18th International Conference on Data Engineering*, Washington, DC, USA, pp. 117–128. IEEE Computer Society.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics 5*(1), 32–38.

Valtchev, P. (1999). *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Thèse d'informatique, Université Grenoble 1.

Summary

Ontology matching amounts to discovering semantic relations (equivalence, subsumption, etc.) between entities of two ontologies. The primary tool for matching ontologies is to measure similarity between pairs of entities. Some efficient matching methods applies a mutual reinforcement principle within a graph of entity pairs. The similarity of a pair of entities depends on the similarity of all neighbor pairs in the graph. Similarity values are computed through an iterative fixed point computation. We present a unifying framework, called structure-guided alignment, which can simulate these methods in spite of numerous technical differences. It combines elements from graph matching and matrix calculus. We finally present the new implementation of the OLA method in this framework.